Security Analysis of VulnBank Application

S. Ioan-Victor, S. Hojda, T. Copaciu

Quality Aspects of Security in Software Testing
QASST Project 2026

Overview

Presentation Agenda

l, Strategy, Exploitation & Remediation

Presentation Contents

- **Target:** VulnBank Application Overview
- **Methodology:** Hybrid Approach (Defensive & Offensive)
- **Findings:** Summary of Identified Vulnerabilities
- **Deep Dive:** Offensive Exploitation (SQL Injection)
- **Resolution:** Remediation Steps & Conclusions

Software Tested

VulnBank Application

Target Application: VulnBank

- **Type:** Intentionally vulnerable banking simulation (Web & API).
- **Critical Features:**
  - Fund Transfers
  - Loan Requests
  - Profile Management
- **Why this target?** It handles sensitive financial data (PII, Transactions), making it an ideal candidate for security testing.

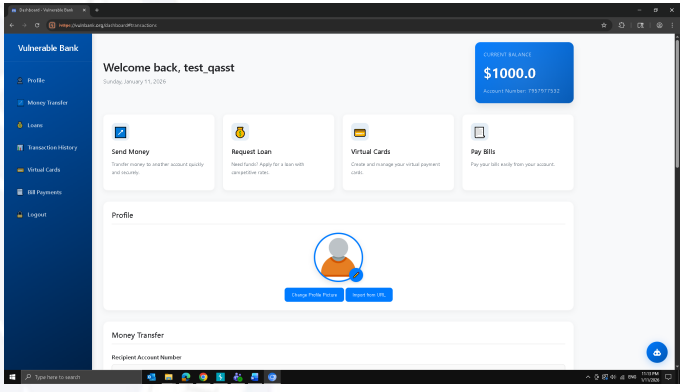


Figure 1: VulnBank Dashboard

Security Approach

Hybrid Strategy

Hybrid Strategy: White-box & Black-box

Defensive (White-box)

- **Tool:** Snyk (SAST)
- **Focus:** Code Quality & Configuration
- **Key Findings:**
  - Insecure Code Patterns (f-strings)
  - SSL Verification Disabled
  - Missing Cookie Attributes

Offensive (Black-box)

- **Tools:** Burp Suite, Nessus (DAST)
- **Focus:** Runtime Exploitation
- **Key Findings:**
- **SQL Injection (API)**
- API Documentation Exposure
- Cleartext Credentials

Critical Findings

Offensive Exploitation

Vulnerability Summary

Defensive Analysis (Snyk):

- SQL Injection in `app.py` (High)
- Improper Certificate Validation (High)
- DOM-based XSS (Medium)

Offensive Analysis :

- (Burp Suite) **Critical SQL Injection in /api/transactions**
- API Documentation Exposure (/api/docs)
- Nessus: PlainText Credentials
- Clickbaiting.

Exploitation: SQL Injection (Manual)

- **Endpoint:** `GET /api/transactions`
- **Method:** Intercepted request via Burp Suite Proxy.
- **Payload:** `' OR '1'='1`
- **Result:** The server returned the transaction history for **ALL** users.

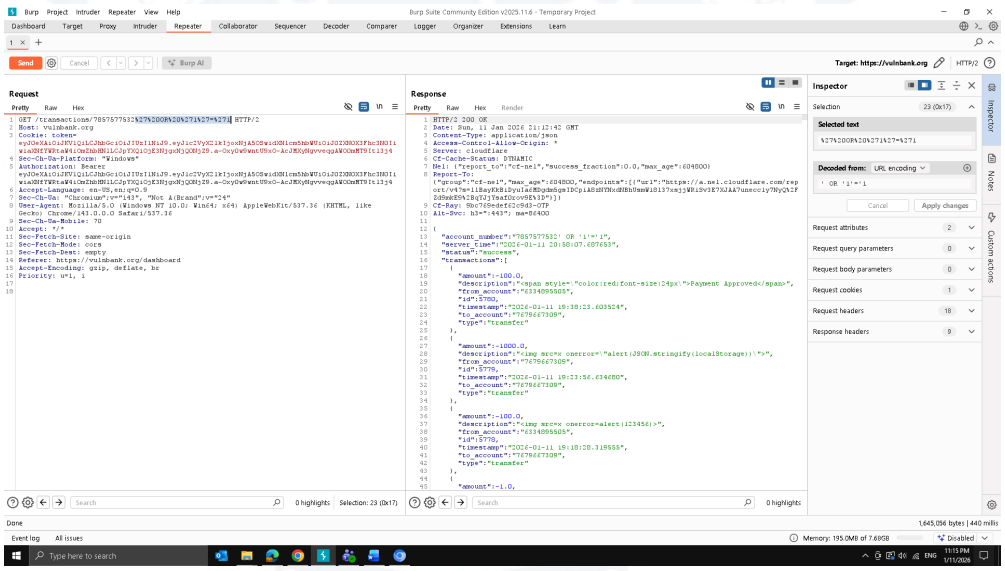


Figure 2: Burp Suite Repeater Proof of Concept

Automatic Plaintext exploit

- Set up MitM
- Intercept.
- Profit

```
[+] INTERCEPTED POST REQUEST TO: vulnbank.org [!] CREDENTIALS
FOUND IN BODY: "username":"suuup dude","password":"sper ca nu
vezi aasta lol lmao that woild be so emberssaing"
-----------------------------------------
```

Clickbaiting exploit

- Load transparent Iframe that contains some logging in info, or anything important that the user doesn't neccesarily want to click.
- Load something unassuming below it
- Profit

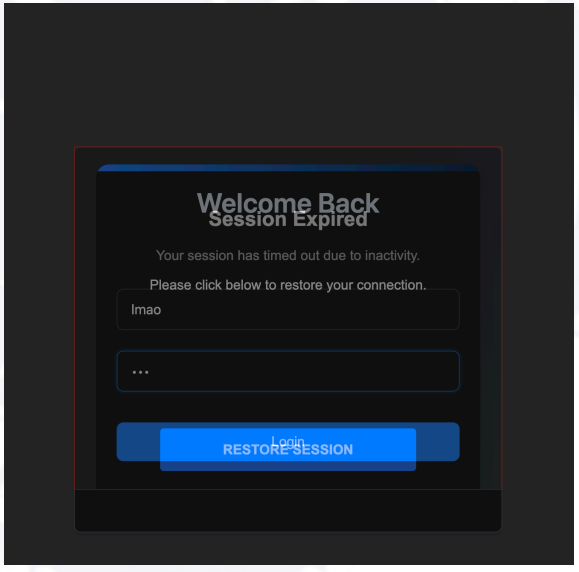


Figure 3: Ghostly iframe

Remediation

Fixing the Issues

Remediation Steps Reporting

1. Fixing SQL Injection

Replaced dynamic SQL concatenation with **Parameterized Queries** (Prepared Statements).

2. Hardening API

Disabled Swagger UI in production environment and enforced Authentication on docs endpoints.

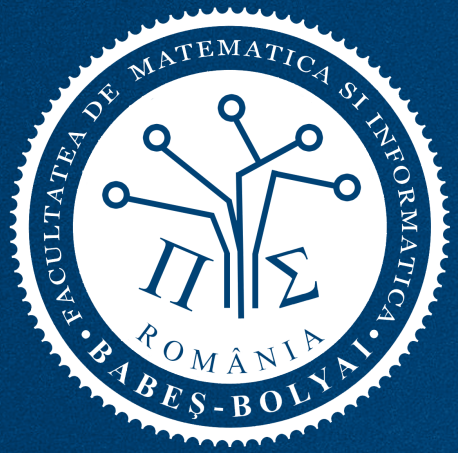
3. Reporting (RIMSEC)

Reported as **High Severity**.

- **R**eproducible: 100% via Burp.
- **I**mpact: Critical Data Leak.
- **M**itigation: Code patch applied.

Conclusions

- **Synergy:** The Defensive approach (SAST) identified potential flaws, while the Offensive approach (DAST) validated their real-world impact.
- **Tooling:** Automated tools (Nessus/Snyk) are excellent for baselines, but manual testing (Burp Suite) is essential for logic flaws like the API SQL Injection.
- **Lesson Learned:** Security must be layered. A simple configuration error (exposed docs) led to a critical data breach.



QUALITY ASPECTS OF SECURITY IN SOFTWARE TESTING
QASST PROJECT 2026

Babeş-Bolyai University

CS Master - CyberSecurity